CLOSE ENOUGH RECOGNITION OF (STEP) FUNCTIONS USING RECOGNITION COEFFICIENTS BASED ON SQUARE WAVES

Abstract: *We present a technique to make a close enough recognition of a function, based on recognition coefficients. The recognition coefficients are determined using inner products of the function to be recognized with a complete set of functions. This is how generalized Fourier coefficients are calculated. However they may not give a series representation of the function to be recognized as the complete set of functions need not be orthogonal.*

In particular step functions are investigated, and formulae derived for recognition coefficients of a step function based on a complete set of square wave functions. These recognition coefficients are simpler faster and more accurate to calculate than coefficients of both regular Fourier series, and series based on square waves.

Introduction

Our objective is to make a close enough recognition of a function, usually a step function which takes only a finite number of values. Measurements may be approximate, or there may be noise in the input, or there may be other changes in the data when it collected on different occasions. Also, if the experiment or process is repeated, there may be variations in the data received. In view of this, it is not possible to make an exact comparison, and we need to make a close enough recognition of such data.

One possibility is to represent such functions by a (generalized) Fourier series [1], and check corresponding coefficients for closeness. These series use orthogonal bases e.g. sine/cosine functions, Walsh functions, Haar functions or other wavelets [1]. An important requirement for making close enough recognition of such functions, is that small changes in the data should cause small changes in the coefficients. So Haar functions or other wavelets are not a good choice, as successive Haar functions and wavelets use smaller and smaller segments of the functions we are trying to recognize. So a small shift in the data can cause a large change in the coefficients. (Haar functions and other wavelets can be used for image compression [3].) A better choice is to use sine/cosine functions or Walsh functions.

The sine/cosine functions are smooth, have a regular form, and use almost all the values of the function we are trying to recognize when calculating coefficients. The Walsh functions are step functions, do not have such a regular form, and use almost all the values of the function we are trying to recognize when calculating coefficients. The regularity and smoothness of the sine/cosine functions are an advantage for the close recognition of smooth functions. The step like form of Haar functions are an advantage for the close recognition of step functions, but their less regular form may be a disadvantage.

The s/c functions (later) are square waves, have a regular form, and use almost all the values of the function we are trying to recognize when calculating coefficients. They form a complete set [4] but are not orthogonal. They can be used to calculate series coefficients [4] but it is no easy task, and is a slower process than calculating standard Fourier series coefficients. The recognition coefficients we calculate are simpler faster and more accurate to calculate than coefficients of both regular Fourier series and series based on square waves. Standard generalized Fourier series coefficient formulae are calculated using inner products, but they provide recognition coefficients, not series coefficients. Recognition coefficients are compared for closeness.

This is the basis of our recognition technique. Furthermore, the uniformity and step like nature of the s/c functions makes them a good choice for the close enough recognition of step functions. They may also work well for smooth functions, but we have not studied this case.

Mathematical Basis of the recognition method

Let $\{c_1, c_2, ..., c_n, ...\}$ be a complete set of functions for $L_2[u, v]$.
Suppose that $f, g \in L_2[u, v]$ and that $f.c_j = g.c_j$ for each j.
Then f=g almost everywhere on [u, v].
(Here "." denotes the inner product of functions, i.e. the integral of their product over the range [u,v].)

Proof:
Since C is a complete set, there is a subset $B \subseteq C$ such that B is a linearly independent complete set. Suppose that $B = \{b_1, b_2, ..., b_m, ...\}$.

As each b_i equals some c_j, and as $f.c_j = g.c_j$, it follows that $f.b_i = g.b_i$ for each i. This can be written as the following "matrix":

$$\begin{bmatrix} b_1 \\ . \\ . \\ b_m \\ . \\ . \end{bmatrix} [\, f \,] = \begin{bmatrix} b_1 \\ . \\ . \\ b_m \\ . \\ . \end{bmatrix} [\, g \,] \qquad (1)$$

By the Gram Schmidt orthonormalization technique, we can find an orthonormal complete set $\{e_1, e_2, ..., e_m, ...\}$ from $\{b_1, b_2, ..., b_m, ...\}$. Careful study of the technique, reveals that e_m is a linear combination of $b_1, b_2, ..., b_m$ where the coefficient of b_m is non zero. We can therefore write:

$$\begin{bmatrix} e_1 \\ . \\ . \\ e_m \\ . \\ . \end{bmatrix} = T \begin{bmatrix} b_1 \\ . \\ . \\ b_m \\ . \\ . \end{bmatrix} \qquad (2)$$

where T is a lower triangular matrix with non zero diagonal elements. (T is therefore invertible.)

By multiplying (1) by T on the left and using associativity and (2) we get:

$$\begin{bmatrix} e_1 \\ . \\ . \\ e_m \\ . \\ . \end{bmatrix} [\, f \,] = \begin{bmatrix} e_1 \\ . \\ . \\ e_m \\ . \\ . \end{bmatrix} [\, g \,] \qquad (3)$$

As $\{e_1, e_2, ..., e_m, ...\}$ is an orthornormal complete set and by (3) $f.e_i = g.e_i$ for each i, it therefore follows that $f = \Sigma_i (f.e_i)e_i = \Sigma_i (g.e_i)e_i = g$.

Q.E.D.

Notes

1) In the above, orthogonality and linear independence are not required.
2) This result generalizes a similar result for an orthogonal complete set.
3) In the following we shall use $f.c_j$, $g.c_j$ as recognition coefficients for comparison purposes when $\{c_1, c_2, ..., c_n, ...\}$ is a complete set of functions.

The square wave functions s, c

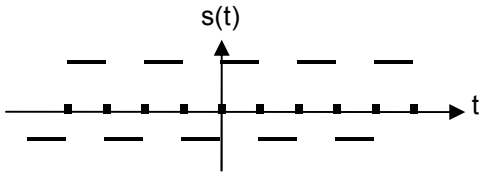
Let t be a real number. The floor of t, written $\lfloor t \rfloor$ is the largest integer less than or equal to t, e.g. $\lfloor 1.5 \rfloor = 1$, $\lfloor -1.5 \rfloor = -2$.

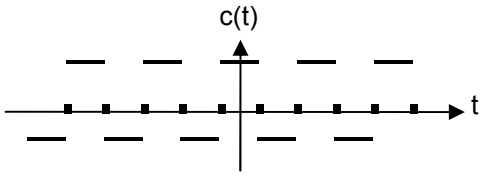
Here are the definitions of the square wave periodic functions s, c.

$s(t) = 0$ if t is an integer
$= +1$ if t is not an integer and $\lfloor t \rfloor$ is even
$= -1$ if t is not an integer and $\lfloor t \rfloor$ is odd

$c(t) = s(t+½)$

The functions s, c have a period of 2 and an amplitude of 1. Apart for a phase shift of ½, they are identical. Their graphs have the following form.





s(-t) = -s(t) and so s is an odd function.

c(-t) = c(t) and so c is an even function.

It is known [4] that { 1, c(nt), s(nt) : $n \geq 1$} is a complete set.

The recognition coefficients of a function f in $L_2[0, 2]$ using these square waves are:

$$a_0 = \int_0^2 1 \times f(t)dt = \int_0^2 f(t)dt.$$

$$a_n = \int_0^2 c(nt)f(t)dt, \qquad b_n = \int_0^2 s(nt)f(t)dt \qquad \text{where } n \geq 1.$$

If f is an odd function then $a_n = 0$ for $n \geq 0$.
If f is an even function then $b_n = 0$ for $n \geq 1$.

Notes

1) All the functions 1, c(nt), s(nt) have norm $\sqrt{2}$, where $n \geq 1$. They are not orthogonal though for $m \geq 1$, $n \geq 1$, c(mt) is orthogonal to s(nt), and 1 is orthogonal to c(mt), s(nt).
2) If a function is defined in the range [0, 1] we can extend it to [-1, 1] as an odd function by defining f(-t)=-f(t), or an even function by defining f(-t)=-f(t). The odd extension will have coefficients a_n=0 and the even extension will have coefficients b_n=0. So by using the appropriate extension, we use either a_n for recognition or b_n for recognition. This has similarities to Fourier sine or cosine series, but we do not discuss this approach further.

The square wave functions s, c avoid a certain difficulty

There is a certain difficulty which should be avoided when using a set of functions for making a recognition. We shall explain this difficulty by an analogy. Suppose we wish to use two linearly independent vectors in the plane for computing recognition coefficients using inner products. Now if these vectors have almost the same direction, or they are almost in opposite directions to each other there may be problems. As can be seen from the two diagrams, P, Q would have similar inner products with u, v, even though they are quite far from each other. This is because taking an inner product corresponds to finding points on u, v, where the perpendiculars from P, Q meet u, v. These points on u, v, will be close to each other for P, Q and the inner products u.P, u.Q would have similar values to v.P, v.Q respectively.

Diagram 1

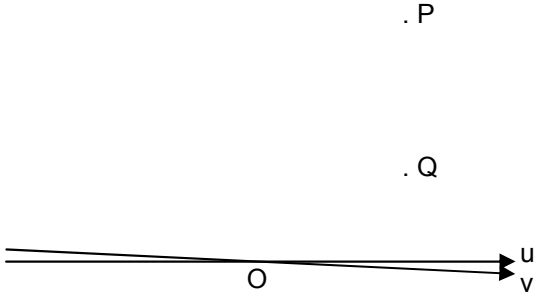
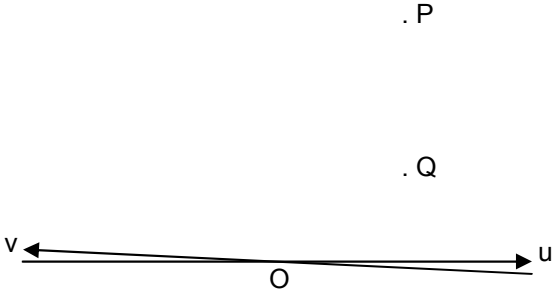


Diagram 2



So in choosing an oblique or non orthogonal coordinate system, it is important that the angle between axes should not be close to 0° or 180°. Equivalently, this means that the cosine of the angle between these axes should not be close to +1 or -1. This cosine can be calculated in terms of inner products and norms and is equal to (u.v) / ||u|| ||v|| where u, v are vectors defining the directions of these axes.

Similarly for the square wave functions we must check that the cosine of the angle between them is not close to +1 or -1. Though Wei and Chen in their paper [4] on square waves make no such check, one of their results enables us to make this check.

Wei and Chen develop square waves like ours but scaled differently. In their formulation the wavelength is 2π and the amplitude is $\pi/4$. Their X, Y functions are related to our s, c functions by $X(t) = (\pi/4)c(\pi t)$ and $Y(t) = (\pi/4)s(\pi t)$.

They work with the complete set of functions { 1, X(nt), Y(nt) : $n \geq 1$} and of course X(mt) is orthogonal to Y(nt) for $m \geq 1$, $n \geq 1$. Also 1 is orthogonal to X(mt), Y(nt) for $m \geq 1$, $n \geq 1$. All the functions X(mt), Y(nt) have norm $\sqrt{(\pi^3/8)}$ for $m \geq 1$, $n \geq 1$.

(In their formulation, the function 1 has norm $\sqrt{2\pi}$.)

It remains for us to determine the cosine of the angle between the functions X(mt), X(nt) and Y(mt), Y(nt) where $m \neq n$ and $m \geq 1$, $n \geq 1$, since all other distinct pairs of functions in the complete set are orthogonal to each other.

Wei and Chen determine the value of the following inner products.

$X(mt).X(nt) = (\pi^3/8)\, A(mn / (m, n)^2)$ and $Y(mt).Y(nt) = (\pi^3/8)\, B(mn / (m, n)^2)$.

Here (m, n) denotes the greatest common divisor of m and n and A(k), B(k) are defined as follows.

$A(k) = (1/k)\,(-1)^{(k-1)/2}$ if k= 1, 3, 5, 7,...
$= 0$ if k= 2, 4, 6, 8,...

$B(k) = (1/k)$ if k= 1, 3, 5, 7,...
$= 0$ if k= 2, 4, 6, 8,...

As all the functions X(mt), X(nt) have norm $\sqrt{(\pi^3/8)}$ for $m \geq 1$, $n \geq 1$, the cosines of the angle between X(mt), X(nt) is $(\pi^3/8)\, A(mn / (m, n)^2) / (\sqrt{(\pi^3/8)}\, \sqrt{(\pi^3/8)}) = A(mn / (m, n)^2)$.

Similarly the cosines of the angle between Y(mt), Y(nt) is $B(mn / (m, n)^2)$.

Since $mn / (m, n)^2 = 1$ if and only if m=n, this means that when $m \neq n$, the only values which $A(mn / (m, n)^2)$ can take are 0 or -1/3 or +1/5 or -1/7 or +1/9 etc.

Similarly when $m \neq n$, the only values which $A(mn / (m, n)^2)$ can take are 0 or 1/3 or 1/5 or 1/7 or +1/9 etc.

Summarizing, these cosines may only take the values 0, $\pm$1/3, +1/5, $\pm$1/7, +1/9 etc. These values are not close to +1 or -1 and this makes it reasonable to use the X, Y functions for recognition purposes. Here is the same situation described in terms of angles.

The cosine 0 corresponds to an angle of 90°.
The cosine +1/3 corresponds to an angle of 71°.
The cosine -1/3 corresponds to an angle of 109°.
The cosine +1/5 corresponds to an angle of 78°.
The cosine +1/7 corresponds to an angle of 82°.
The cosine -1/7 corresponds to an angle of 98°.
The cosine +1/9 corresponds to an angle of 84°.
Etc.

So we see that these angles differ by at most 19° from 90°, i.e. from being orthogonal.

Since the s, c functions are like the X, Y functions but just scaled differently, this also means that the cosines and the angles between the s, c functions may only take the above values. This also makes it reasonable to use the s, c functions for recognition purposes.

The integrals of s, c are C, S respectively

Let us define $C(T) = \int_0^T s(t)dt$.

$C(T) = T - \lfloor T \rfloor$ when $\lfloor T \rfloor$ is even
$= 1 - (T - \lfloor T \rfloor)$ when $\lfloor T \rfloor$ is odd

Since s(t) is an odd function, C(-T) = C(T).

Let us define $S(T) = \int_0^T c(t)dt$.

$S(T) = \int_0^T s(t+½)dt$ since c(t) = s(t+½),

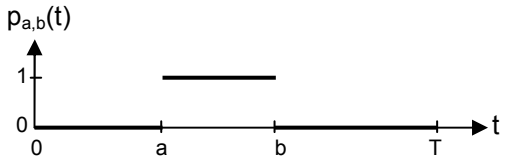
$= [C(t+½)]_0^T = C(T) - C(½) = C(T) - ½$.

Since c(t) is an even function, S(-T) = -S(T).

Note that regarding the indefinite integral, -½ can be absorbed into the constant of integration. Therefore $\int c(t)dt = C(t+½)$ and $\int s(t)dt = C(t)$.

Recognition coefficients for a single pulse or step of unit height

Consider the function $p_{a,b}(t)$ where t ranges over the interval [0,T] and $0 \le a \le b \le T$.



We first calculate recognition coefficients assuming T=2, and then deal with the general case. In the interval [0,T], the value of $p_{a,b}(t)$ is 1 only when t lies between a and b, otherwise it is zero. So we need only integrate over [a, b], the range where $p_{a,b}(t)$ is non zero.

Therefore:

$a_0 = \int_a^b p_{a,b}(t)dt = \int_a^b 1dt = (b-a)$.

$a_n = \int_b^a 1 \times c(nt)dt = [S(nt)/n]_a^b = (S(nb) - S(na)) / n$, where $n \ge 1$.

$b_n = \int_b^a 1 \times s(nt)dt = [C(nt)/n]_a^b = (C(nb) - C(na)) / n$, where $n \ge 1$.

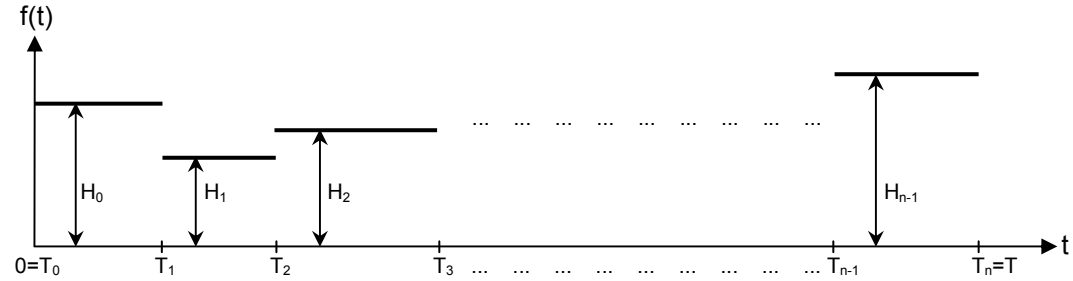
In the general case, t which ranges over the interval [0, T] can be scaled to t' which ranges over the range [0,2], by substituting t' = (2/T)t. When t = a, b respectively t'=(2/T)a, (2/T)b respectively. So we calculate recognition coefficients for $p_{(2/T)a,\ (2/T)b}(t)$ using the above to obtain the following recognition coefficients for any T>0.

$a_0 = (2/T)(b-a)$.
$a_n = (S(2nb/T) - S(2na/T)) / n$, where $n \ge 1$.
$b_n = (C(2nb/T) - C(2na/T)) / n$, where $n \ge 1$.

Recognition coefficients for a finite number of steps

Suppose that $0=T_0<T_1<T_2... <T_n=T$ and that $f(t) = \sum_{i=0}^{n-1} H_i p_{T_i,\ T_{i+1}}(t)$.

The graph of f(t) has the following form.



By summing the recognition coefficients $H_i p_{T_i, T_{i+1}}(t)$ over i, we obtain the following recognition coefficients for f(t), for any T>0.

$$a_0 = (2/T) \sum_{i=0}^{n-1} H_i \, (T_{i+1} - T_i).$$

$$a_n = (1/n) \sum_{i=0}^{n-1} H_i \, (S(2nT_{i+1}/T) - S(2nT_i/T)), \quad \text{where } n \geq 1.$$

$$b_n = (1/n) \sum_{i=0}^{n-1} H_i \, (C(2nT_{i+1}/T) - C(2nT_i/T)), \quad \text{where } n \geq 1.$$

Advantages of using s, c, C, S over sine, cosine

The computation of the functions s, c, C, S is simple, fast, and exact apart from rounding errors. The functions sine, cosine need to be approximated by polynomials and there would be both approximation and rounding errors, as well as slower speed of computation. Also, recognizing step functions by means of sine, cosine may require the computation of more coefficients, in view of the rather different forms of these functions. All in all, there appear to be several advantages to using s, c, C, S as opposed to sine, cosine for recognizing step functions.

Keystroke dynamics recognition using our technique

Keystroke dynamics are also called typing rhythms and a survey of this topic is given in [5]. The specific keystroke dynamics problem we study is to recognize the user from the dynamics of a characteristic twiddle of his fingers on the keyboard. The user is asked to twiddle his fingers on the keyboard, using one or both hands, and may press several keys simultaneously. Several such samples are collected from the user initially, where the user repeats as best as he can his twiddle. What we have tried to develop is a kind of keyboard signature, which takes into account the form and timing of the user's characteristic twiddle. This would be used as a replacement for a password, when the user logs on to the computer. We do not propose a solution to the problem continuously monitoring and recognizing a user from his normal usage of the keyboard.

For each sample collected a note is made of which keys were pressed or released and the time that this occurred. For each sample the time is scaled so that the time of the first key being pressed is 0 and the time of the last key being released is 2.(Any positive constant can be used in place of 2.) Determine the (x, y) coordinates of the keys pressed, and the number k of keys pressed. If several keys are pressed simultaneously at a given moment, use the average of their coordinates. Regarding the coordinate system, oblique or rectangular coordinates may be used, but choose coordinates in a way so that no key has a zero coordinate. We can therefore use (0, 0) to represent that no key is depressed.

In this way we obtain for each sample, three step functions x(t), y(t), k(t) giving the average coordinate x(t) of the keys pressed at time t, the average coordinate y(t) of the keys pressed at time t, and the number of keys k(t) pressed at time t. So if s such samples are collected, we determine average step functions for these samples as follows.

$$\bar{x}(t) = [x_1(t)+...+ x_s(t)] / s, \qquad \bar{y}(t) = [y_1(t)+...+ y_s(t)] / s, \qquad \bar{k}(t) = [k_1(t)+...+ k_s(t)] / s.$$

The average step functions $\bar{x}(t)$, $\bar{y}(t)$, $\bar{k}(t)$ are used as a reference for comparison purposes.

Now if we wish to recognize a user by the twiddling of his fingers on the keyboard, we collect a sample of his twiddling on the keyboard and determine x(t), y(t), k(t) for this sample. Then calculate and compare sufficiently many recognition coefficients of x(t) with $\bar{x}(t)$, y(t) with $\bar{y}(t)$, and k(t) with $\bar{k}(t)$, for closeness.

Equivalently, we could calculate sufficiently many recognition coefficients of $x_i(t)$, $y_i(t)$, $k_i(t)$ for i=1,...,s. Then average corresponding recognition coefficients obtaining recognition coefficients of $\bar{x}(t)$, $\bar{y}(t)$, $\bar{k}(t)$ and continue as before.

Notes

1) As times are scaled to be in the range [0, 2], with 0 being the time of the first key pressed and 2 being the time of the last key released, the user does not need to repeat his twiddle in the same time frame. This means that if we wish to check this aspect too, an explicit check must be added.

2) If we wish allow the user to repeat his movement anywhere on the keyboard, we do not compare the a_0 coefficients of x(t), y(t) with $\bar{x}(t)$, $\bar{y}(t)$ respectively, since the a_0 coefficients of these functions give the average values of x(t), y(t), $\bar{x}(t)$, $\bar{y}(t)$ for t in the range [0, 2] i.e. they determine average positions for t in the range [0, 2].

The use of moments for recognition

Moments may also be used for recognition purposes [2]. Indeed, this is equivalent to using the non orthogonal complete set of functions $\{ x^n : n \geq 0\}$, where we scaled the data so that x ranges over the the interval [0,1]. Here are some problems with using this complete set.

1) These functions do not avoid the difficulty mentioned earlier that the cosine of the angle between them is not close to +1 or -1. The cosine of the angle between x^m and x^n is $\sqrt{(2n+1)}\sqrt{(2m+1)}/(n+m+1)$. This cosine is 0.995 when m=5 and n=6, and is 0.999 when m=10 and n=11.
2) These functions give a greater weight to the data near the value 1. Even if the data is scaled to range over [-1, +1], greater weight is given to data near the two ends of the interval, and the data towards the middle are given lower weights.
3) Underflow can occur when computing x^n. Overflow is avoided by scaling the data as above.

On the other hand, the use of moments has advantages. Approximation and rounding errors can occur in the computation of sin, cos but the computation of x^n is exact apart from rounding errors. The computation of moments is faster than the computation of Fourier coefficients and our recognition coefficients, but careful programming is required to achieve this.

Comparison and conclusions

We have compared the use of our recognition technique with the established recognition technique based on Fourier series coefficients, using the keystroke dynamics recognition problem for making this comparison. Though we have not succeeded in solving the keystroke dynamics problem, we obtained similar results with our recognition technique, and with the established technique of using Fourier series coefficients. By suitably adjusting the "grade of agreement" and "number of standard deviations for closeness" parameters we could achieve a low false accept rate but a high false reject rate, or a low false reject rate and a high false accept rate with both methods. We were unable to achieve a low false accept rate and false reject rate with both methods.

Regarding execution speed, using square wave recognition coefficients was faster than using standard Fourier series coefficients. In a comparison of speeds of computing the basic functions used in the recognition process i.e. sin, cos compared with S, C used for square wave recognition, computation of S, C was about 75% faster than computing sin, cos. Coding S, C in assembler, can further improve the execution speed of these functions.

Regarding accuracy, the functions sin, cos are approximated by polynomials whereas there are exact formulae for S, C.

While we have not solved the keystroke dynamics recognition problem, this work demonstrates that recognition coefficients based on square waves, performed as well as the established technique of using Fourier series coefficients. In view of significantly faster computation speed and improved accuracy of computation, this new technique may be useful in efficiently solving other recognition problems.

We also tried using moments to solve the keystroke dynamics problem. Surprisingly, we got

similar results to those obtained using Fourier series coefficients and our recognition coefficients. Perhaps this is because we did not calculate moments for large n, so that the problems we listed in the previous section were not acute. On the other hand, if moments need to be calculated for large n, their use should be avoided.

Acknowledgments
Thanks to S. Engelberg and P. Morginstin for their help.

References

[1] "*Fourier and wavelet analysis*", G. Bachman, L. Narici and E. Beckenstein, Springer 2000.
[2] "*Digital Image Processing*", K. R. Castleman, 1996.
[3] "*Data Compression: The Complete Reference*", D. Salomon, Springer, 3rd ed., 2004.
[4] "Square Wave Analysis", Y. Wei and N. Chen, Journal of Mathematical Physics, Vol. 39 No. 8 pp. 4226-4245, August 1998.
[5] "Keystroke dynamics & corporate security", J. C. Checco, Ticker, Wall street technology association, www.wsta.org/publications, September/October 2003.

R.B. Yehezkael (formerly Haskell).
Revised August 2006 - אב תשס"ו.
Jerusalem College of Technology - Machon Lev,
Hawaad Haleumi 21, Jerusalem 91160, ISRAEL.
Tel: 02-6751111 e-mail: rafi@jct.ac.il
This article can be found on my internet page
http://www.cc.jct.ac.il/~rafi